Fantasy Hockey Auction Draft Strategy

A Project on Solving Fantasy Hockey Auction Drafts

Problem Statement and Motivation

- **Objective:** Identify an optimal strategy for fantasy hockey auction drafts.
- **Game Structure:**
  - **Players:** n players draft a team of:
    - o forwards
    - d defensemen
    - g goalies
  - **Budget:** Each player has a fixed budget of b.
  - **Auction Format:**
    - English auction—each round:
      - Players **raise** (bid increases by 1) or **fold** (exit the round).
      - The last remaining player wins the athlete at their bid.
  - **Constraints:** Budget management and team composition rules.
- **Assumptions:** All athlete values are known beforehand.
- **Motivation:** Inspired by my personal yearly fantasy hockey auction league—an engaging and strategic game.

Success Metrics

- **Qualitative Success:**
  - Professor couldn't beat the algorithm in direct play
  - Expert players find it challenging to exploit
- **Live Demo Planned:**
  - Will demonstrate real-time bidding decisions during presentation
  - Audience can challenge the algorithm
- **Note on Baselines:**
  - Limited existing baselines to compare against
  - Existing automated strategies (like ESPN's) are highly exploitable

Literature Review

- **Limited Direct Work in Fantasy Hockey Optimization:**
  - No academic literature specifically on auction fantasy hockey strategy
- **Current Industry Approaches:**
  - ESPN Fantasy Draft: Uses analyst-suggested values with hard-coded bidding caps
  - These approaches are simplistic and easily exploitable
- **Related Game Theory Problems:**
  - Colonel Blotto games: Similar resource allocation dynamics
  - Sequential auction theory: Provides some theoretical foundations
  - These simpler models can use linear programming, but don't scale to our complex scenario
- **Novel Application Area:**
  - Bridging reinforcement learning and auction theory in complex, sequential settings

Approach 1: AlphaZero Method

Overview

- Leverages self-play and Monte Carlo Tree Search (MCTS).
- Operates in a **perfect information** setting. Every piece of information is visible at each turn.
- **Auction Dynamics:**
  - In each round, players choose to **raise** or **fold**.
  - A raise increases the bid by 1; folding exits the round.
  - The last remaining player wins the athlete at their bid.

Approach 1: AlphaZero Method

Mathematical Formulation

For a given game state (s):

$$\pi^*(s) = \arg\max_{a \in \{\text{raise}, \text{fold}\}} Q(s, a)$$

- $\pi^*(s)$: The ideal policy function we try to fit to during self-play and $Q(s, a)$ is the value of playing a given state s.

Approach 1: AlphaZero Method

Challenges

- **Enormous Search Space:**
    - With n players and budget b, the number of possible turns is huge.
    - The game tree complexity grows as approximately $O(2^{nb})$ (compared to 60–80 moves in chess).
- **Computational Overhead:**
  MCTS becomes expensive as the game tree expands.

Approach 2: PPO (Proximal Policy Optimization)

Overview & How It Works

- An actor-critic policy gradient method. We directly optimize our policy to maximize rewards.
- Uses a **clipped surrogate objective** to ensure stable policy updates. It's a hacky solution that empirically works well.
- **Auction Modeling:**
  - Each round is approximated as a Vickrey auction (second-price auction).
  - Bids are modeled as a percentage of the remaining budget.
  - Our model predicts a target mean and variance for the action it wants to play and then we moment match it to an approporiate beta distribution.

Approach 2: PPO

Mathematical Optimization Decision

- **Surrogate Objective:**

$$L^{CLIP}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)A_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\big)\Big]$$

  where:

  - $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
  - A_t is the advantage estimate.
  - ϵ is a small hyperparameter.

- **Bid Modeling:**
  The bid is given by:

$$\text{bid} = b_r \cdot x, \quad x \sim \text{Beta}(\alpha, \beta)$$

  where b_r is the maximum bid we can make, and α, β are generated from mean, variance outputs of our policy network.

Approach 2: PPO

Challenges

- **Credit Assignment:**
  Early moves have a larger impact on the final outcome, but PPO's structure treats all moves similarly - in fact discounts earlier moves.
- **Imperfect Information:**
  The auction's uncertainty complicates training. A beta distribution is unimodal. It is unclear that constraint is appropriate for finding a Nash equlibrium.

Approach 3: Deep CFR (Counterfactual Regret Minimization)

Overview & How It Works

- Uses regret minimization to approach Nash equilibria in imperfect information games (will explain on next slide).
- **Key Distinction:**
  Instead of directly predicting regret, our network predicts the **expected reward** $Q(s, a)$ for each action a at state s.
- **Auction Modeling:**
  - Discretize the action space as percentages of the remaining budget (0% to 100%).

Approach 3: Deep CFR

Revised Mathematical Formulation

- **Expected Reward Prediction:**
  For each state s and action a:

$$Q(s,a) = \text{Expected Reward given action } a$$

- **Regret Calculation:**
  Compute regret relative to the expected value of the current policy $\pi(s)$:

$$R(s,a) = Q(s,a) - \sum_{a'} \pi(s,a')Q(s,a')$$

- **Policy Update via Regret Matching:**
  Play the policy based on positive regrets:

$$\sigma(s,a) = \frac{\max(R(s,a),0)}{\sum_{a'} \max(R(s,a'),0)}$$

Approach 3: Deep CFR

Neural Network Approximation & Special Notes

- **Network Optimization:**
  Train the network to approximate $Q(s,a)$ by minimizing:

$$\min_{\theta} \mathbb{E}_{s\sim\mathcal{D}} \left[\sum_{a} \left(Q(s,a) - \hat{Q}(s,a;\theta) \right)^2 \right]$$

  where $\hat{Q}(s,a;\theta)$ is the network's estimate.

- **Implementation Notes:**
  - Start with a 2-player scenario to leverage stronger theoretical guarantees.
  - Adjustments to the original Deep CFR formulation improve practical performance.
  - The athlete pool is fixed per season—requiring retraining with updated projections.
  - We train on all samples we have seen across all iterations.

Approach 3: Deep CFR

Policy Network Training with JSD Minimization

- **Additional Training:**
  - Train a separate policy network to capture the aggregate strategy.
- **Jensen-Shannon Divergence Minimization:**
  - **Objective:** Align the network's predicted distribution $\pi_\phi(s)$ with the aggregate policy $\pi_{\text{agg}}(s)$ observed across all iterations.
  - **Formulation:**

$$\min_\phi D_{JS}\Big(\pi_{\text{agg}}(s) \parallel \pi_\phi(s)\Big)$$

  - **Benefit:**
    - Encourages the policy network to learn from the best strategies played so far.
    - Provides a robust policy approximation that generalizes across iterations.
    - Jenson-Shannon over KL-Divergence gave more stable outputs.

Approach 3: Deep CFR: Simplified Game Iteration 1

Final Means: `[300, 295]`, **Final Budgets:** `[8, 3]`

Value	Price	Winner	Bids	Curr. Budgets
115	60.0	1	[60, 69]	[100, 40]
110	37.0	0	[69, 37]	[63, 40]
80	37.0	0	[59, 37]	[26, 40]
75	24.0	1	[24, 37]	[26, 16]
50	12.0	0	[24, 12]	[14, 16]
45	0.0	1	[0, 13]	[14, 16]
40	13.0	1	[13, 15]	[14, 3]
35	3.0	0	[13, 3]	[11, 3]
25	3.0	0	[11, 3]	[8, 3]
20	0.0	1	[0, 3]	[8, 3]

Approach 3: Deep CFR: Simplified Game Iteration 3

Final Means: `[325, 270]`, **Final Budgets:** `[29, 8]`

Value	Price	Winner	Bids	Curr. Budgets
115	47.0	0	[88, 47]	[53, 100]
110	50.0	1	[50, 60]	[53, 50]
80	42.0	1	[42, 47]	[53, 8]
75	6.0	0	[50, 6]	[47, 8]
50	6.0	0	[35, 6]	[41, 8]
45	6.0	0	[40, 6]	[35, 8]
40	6.0	0	[35, 6]	[29, 8]
35	0.0	1	[0, 6]	[29, 8]
25	0.0	1	[0, 3]	[29, 8]
20	0.0	1	[0, 8]	[29, 8]

Approach 3: Deep CFR: Simplified Game Iteration 25

Final Means: `[310, 285]`, **Final Budgets:** `[5, 1]`

Value	Price	Winner	Bids	Curr. Budgets
115	61.0	1	[61, 70]	[100, 39]
110	36.0	0	[87, 36]	[64, 39]
80	36.0	0	[61, 36]	[28, 39]
75	26.0	1	[26, 36]	[28, 13]
50	11.0	0	[26, 11]	[17, 13]
45	11.0	0	[16, 11]	[6, 13]
40	6.0	1	[6, 11]	[6, 7]
35	6.0	1	[6, 6]	[6, 1]
25	1.0	0	[6, 1]	[5, 1]
20	0.0	1	[0, 1]	[5, 1]

Approach 3: Deep CFR: Simplified Game Iteration 175

Final Means: `[300, 295]`, **Final Budgets:** `[0, 5]`

Value	Price	Winner	Bids	Curr. Budgets
115	51.0	0	[58, 51]	[49, 100]
110	46.0	1	[46, 70]	[49, 54]
80	31.0	1	[31, 33]	[49, 23]
75	21.0	0	[46, 21]	[28, 23]
50	18.0	0	[26, 18]	[10, 23]
45	9.0	1	[9, 21]	[10, 14]
40	9.0	1	[9, 13]	[10, 5]
35	5.0	0	[9, 5]	[5, 5]
25	5.0	0	[5, 5]	[0, 5]
20	0.0	1	[0, 5]	[0, 5]

Training & Tuning Approach

- **Hyperparameter Tuning Process:**
  - Informal monitoring approach due to computational constraints
  - Regular inspection of performance during training
  - Adjustments made based on observed behaviors
- **Key Modifications:**
  - Custom sampling procedure - each sample gets `weight = np.random.uniform() ** (100000 / self.sample_num)`
  - Modified from literature approach to improve stability and increase converging speed
  - Batch size adjustments to handle inherent noise in the problem
- **Training Logistics:**
  - Long training times prevented formal grid/random search
  - Focus on key parameters with highest expected impact

Implementation & Computational Challenges

- **Implementation Details:**
  - Primary framework: PyTorch for neural network components
  - Custom game environment implementation
- **Computational Bottlenecks:**
  - Self-play sample generation extremely time-consuming
  - Model evaluation requiring many game iterations
- **Adaptations to Make Training Tractable:**
  - Parallelized sample creation pipeline
  - Enabled scaling from 2-player to n-player games
  - Simplified state representations where possible

N-Player Game in Action

🏒 AUCTION IN PROGRESS

Current Player Auction Forward (Value: 358)

Player	Budget	Team Value	F Needed	D Needed	G Needed
Player 0 (You)	67	392	5	4	2
Player 1	100	0	6	4	2
Player 2	40	520	5	4	2
Player 3	60	441	5	4	2

Remaining Players

Position	Players Left	Values
Forwards	20	357, 355, 350, 349, 346, 340, 328, 326, 324, 323, 315, 311, 300, 295, 292, 290, 289, 288, 281, 278
Defensemen	16	307, 260, 259, 251, 250, 243, 242, 239, 233, 229, 225, 218, 206, 202, 196, 191
Goalies	8	273, 272, 266, 265, 260, 252, 240, 193

Recent Auction History

Player	Winner	Price	Bids
392 (F)	You	33	0:35, 1:12, 2:30, 3:33
441 (F)	Player 3	40	0:30, 1:33, 2:40, 3:45
520 (F)	Player 2	60	0:60, 1:53, 2:69, 3:59

Project Evolution

- **Initial vs. Final Approach:**
  - Started with ambitious universal strategy model
  - Narrowed to pre-set game scenarios (fixed players and structure)
  - Created pipeline for specific drafts rather than universal solution
- **Simplification Process:**
  - Recognized game complexity was higher than anticipated
  - Made theoretical simplifications without losing core challenge
- **AI Tool Usage:**
  - Leveraged for generating boilerplate code (and this deck)
  - Helpful for refactoring code sections quickly
  - Especially useful for clearly defined coding tasks
  - Limited use for core algorithm design (required human expertise)

Key Insights & Learnings

- **Surprising Results:**
  - Core game theory principles highly predictive of performance
  - CFR approach with strong theoretical guarantees transferred well to practice
  - Neural networks effective as function approximators in this domain
- **Critical Course Concepts:**
  - Batching importance: Previously underestimated, critical for this noisy problem
  - Learned to use larger batch sizes to stabilize training
- **Future Work (If Given Two More Weeks):**
  - Build clearer web UI to simulate drafts better vs. terminal GUI
  - Improve visualization of strategy development
- **Approach Changes (If Restarting):**
  - Would start with CFR on simplified game version from beginning
  - Avoid starting with most complex version (AlphaZero)

Conclusion

- **Summary:**
  - Explored advanced methods: AlphaZero, PPO, and Deep CFR.
  - Each approach addresses unique aspects of the fantasy hockey auction challenge.
  - Deep CFR emerged as most effective for this imperfect information game.
- **Looking Ahead:**
  - Continuous refinement and experimental validation will bring us closer to an optimal bidding strategy.
  - Open to feedback and further discussion on these methods.

LIVE DEMO

Let's challenge the algorithm to see it in action!

- **Game Parameters:**
  - Budget: $100
  - Players: 2-4
  - Draft order: Predefined
- **Guidelines for Challengers:**
  - Raise or fold each round
  - Strategy objective: Maximize total player value within budget
- **Who can beat the AI?**